Signal Processing Basics using Scilab (Signals and Systems using Scilab)

Presentation · May 2019

DOI: 10.13140/RG.2.2.17057.25441

1 author:

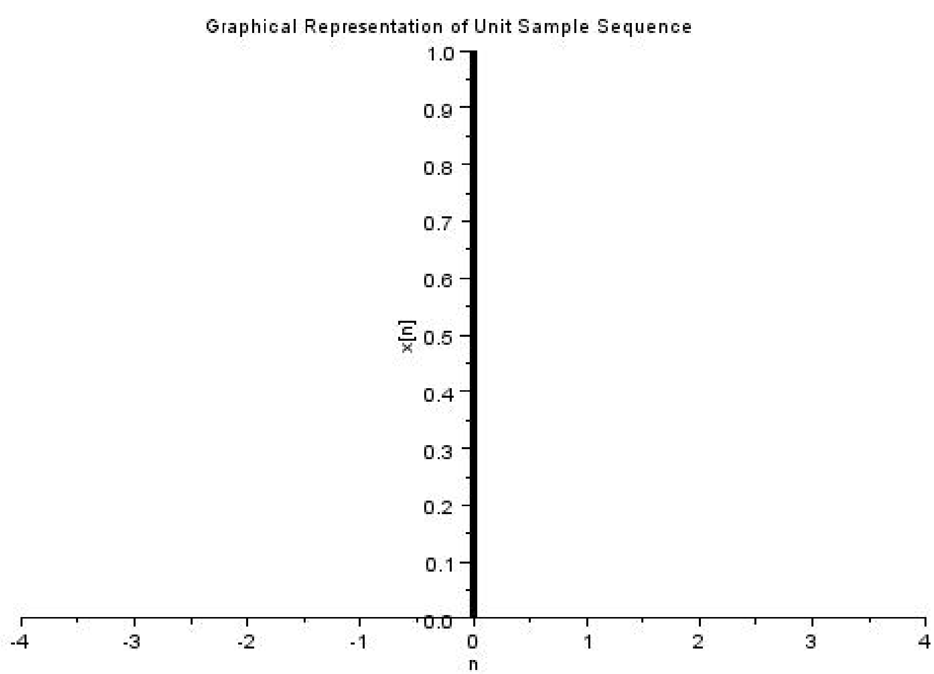
Dr R Senthilkumar
Government College of Engineering Erode

Signal Processing Basics using Scilab (Signals and Systems using Scilab)

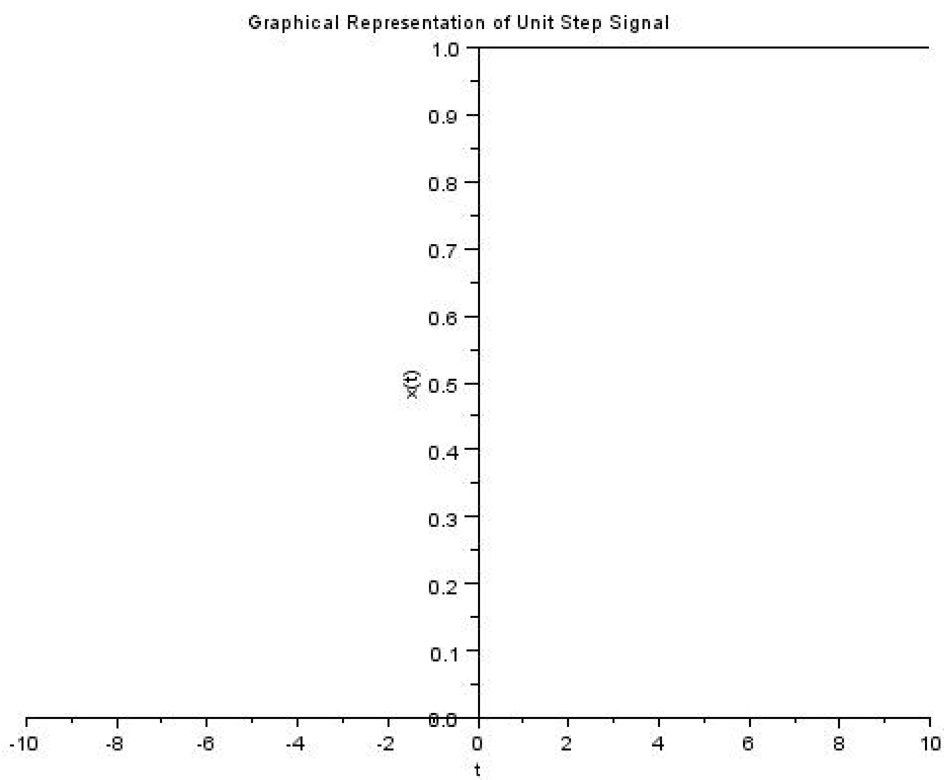
Reference Book: Signals and Systems by R.Senthilkumar, Anuradha Publications, 2nd Edition

Tutorial 1

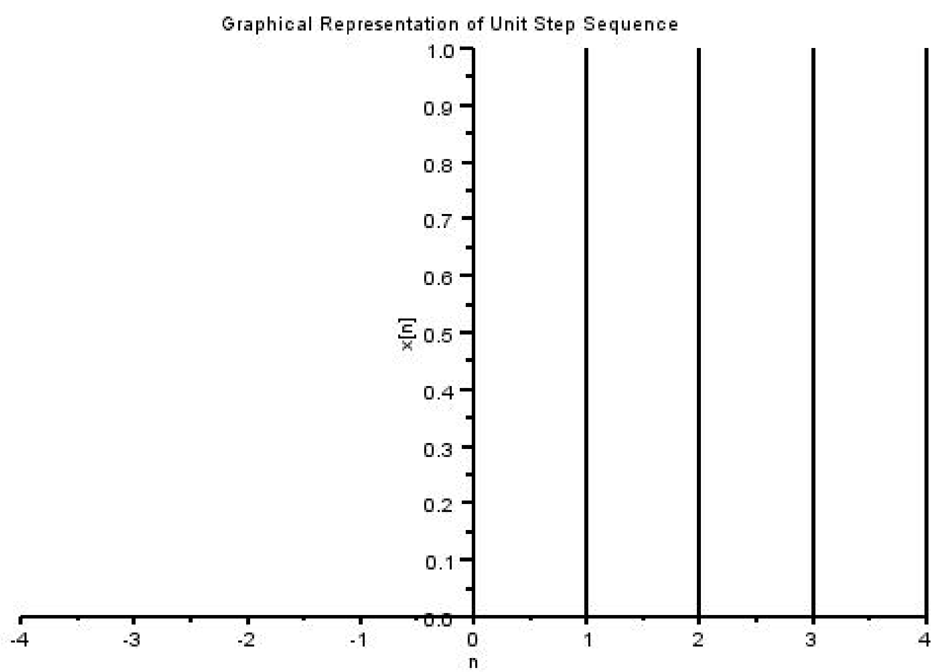
```
//Caption: Program to plot the unit sample sequence
clear all;
clc;
close;
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),1,zeros(1,L)];
b = gca();
b.y_location = "middle";
plot2d3('gnn',n,x)
a=gce();
a.children(1).thickness =4;
xtitle('Graphical Representation of Unit Sample Sequence','n','x[n]');
```

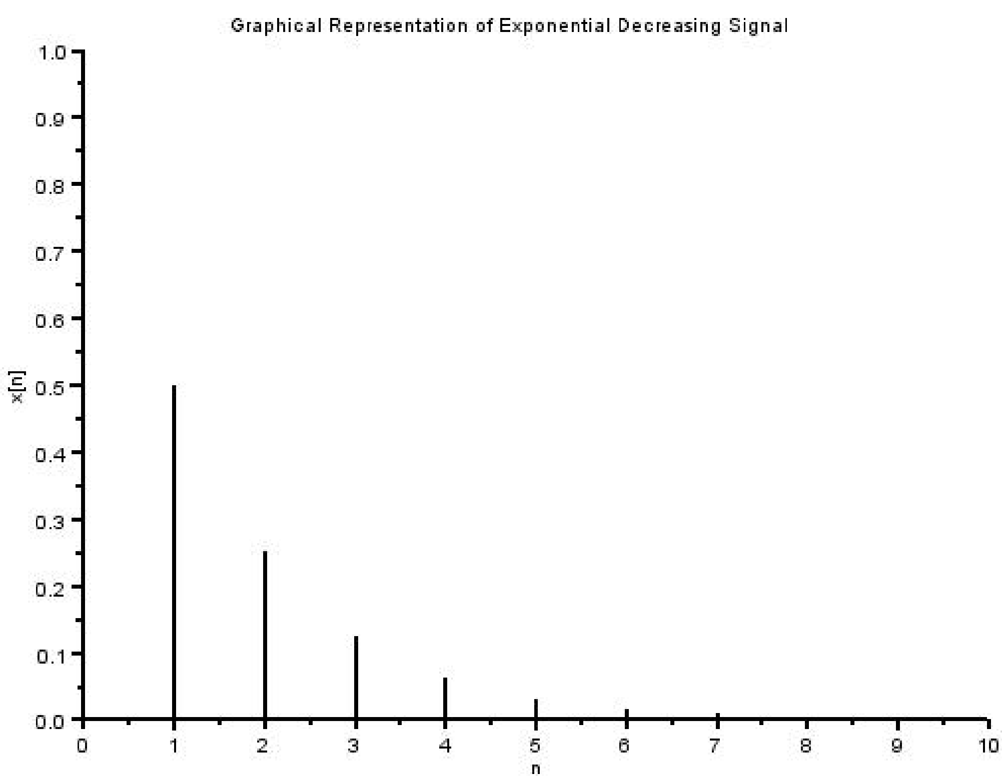
```
//Caption: Program to plot the unit step signal

clear all;
clc;
close;
L = 10;  //Upperlimit
n = -L:L;
x = [zeros(1,L),ones(1,L+1)];
a=gca();
a.y_location = "middle";
plot2d2(n,x)
xtitle('Graphical Representation of Unit Step Signal','t','x(t)');
```

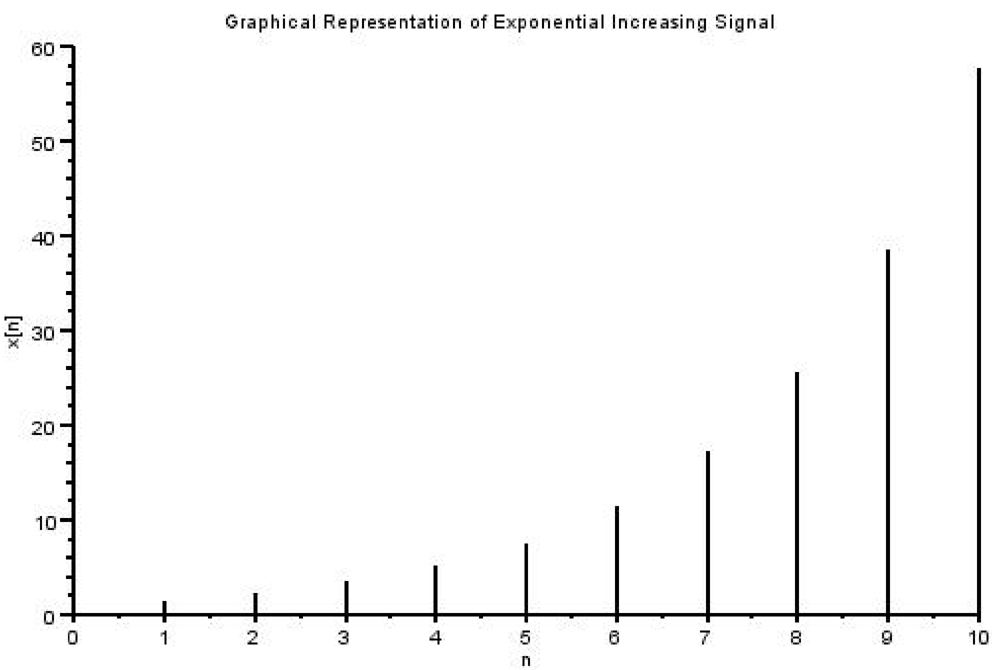
```
//Caption: Program to plot the unit step sequence
clear all;
clc;
close;
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),ones(1,L+1)];
a=gca();
a.thickness = 2;
a.y_location = "middle";
plot2d3('gnn',n,x)
xtitle('Graphical Representation of Unit Step Sequence','n','x[n]');
```



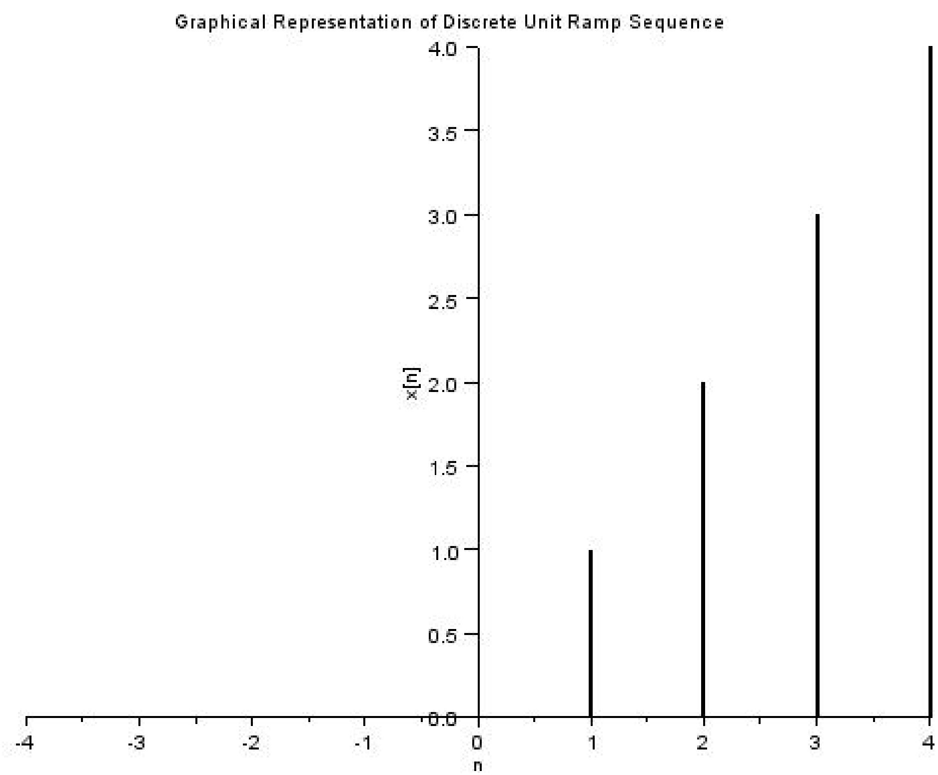
```
//Caption:Program to Plot the discrete exponentially decreasing sequence
// a < 1
clear all;
clc;
a =0.5;
n = 0:10;
x = (a)^n;
a=gca();
a.thickness = 2;
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,x)
xtitle('Graphical Representation of Exponential Decreasing Signal','n','x[n]');
```

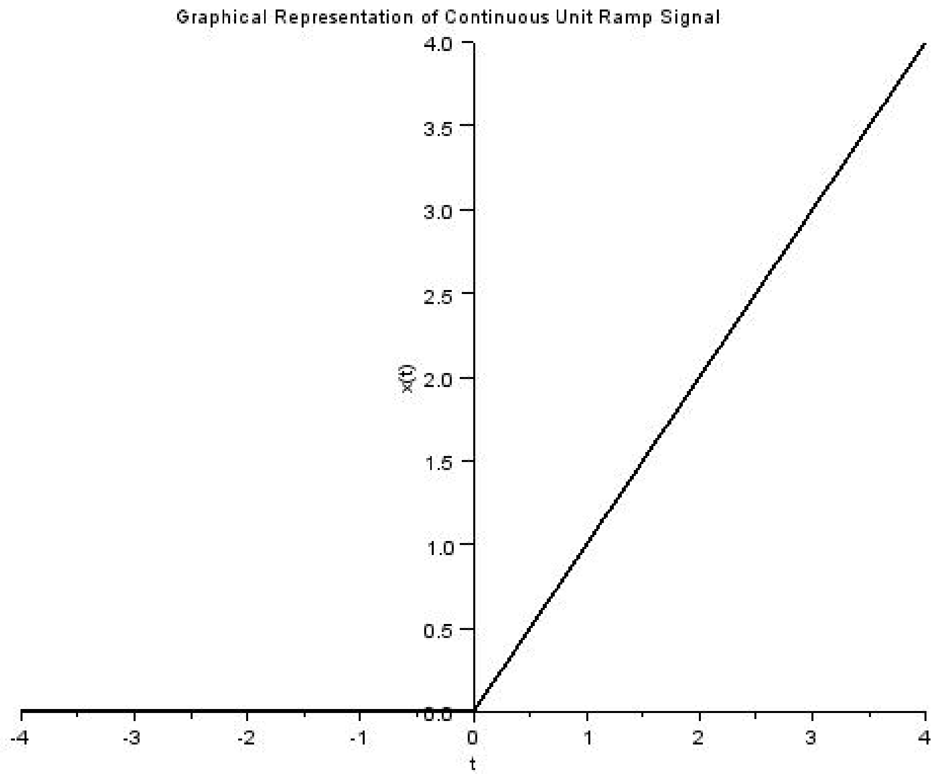
```
//Caption:Program to Plot the discrete exponentially Increasing sequence
// a > 1
clear all;
clc;
a =1.5;
n = 0:10;
x = (a)^n;
a=gca();
a.thickness = 2;
a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',n,x)
xtitle('Graphical Representation of Exponential Increasing Signal','n','x[n]');
```



```
//Caption: Graphical representation of Discrete Time Ramp Sequence
clear;
clc;
close;
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),0:L];
b = gca();
b.y_location = 'middle';
plot2d3('gnn',n,x)
a=gce();
a.children(1).thickness =2;
xtitle('Graphical Representation of Discrete Unit Ramp Sequence','n','x[n]');
```

```
//Caption: Graphical representation of Continuous Time Ramp Signal
clear;
clc;
close;
L = 4;  //Upperlimit
n = -L:L;
x = [zeros(1,L),0:L];
b = gca();
b.y_location = 'middle';
plot2d(n,x)
a=gce();
a.children(1).thickness =2;
xtitle('Graphical Representation of Continuous Unit Ramp Signal','t','x(t)');
```



Graphical Representation of Continuous Unit Ramp Signal
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
x(t)
-4
-3
-2
-1
0
1
2
3
4
t

Tutorial 2

```
//Caption: Program to find the Linear Convolution of two given Sequences
clear all;
clc;
x = [1,2,1,1];
N1 = length(x);
h = [1,1,2,1];
N2 = length(h)
N = N1+N2-1;
y = zeros(1,N);
//Method 1
for i = N1+1:N
  h(i) = 0;
end
for i = N2+1:N
   x(i) = 0;
end
for n = 1:N
 for k = 1:N
   if(n >= k)
     y(n) = y(n)+x(n-k+1)*h(k);
   end
 end
end
disp(y, ‘Linear Convolution Result in Method1 is=')
//Method2
y2 = convol(x,h)
disp(y2,‘Linear Convolution Result in Method2 is=’)
```

Tutorial 3

```
//Caption: Program to find the Autocorrelation of a given Input Sequence
clear all;
clc;
x = input('Enter the given discrete time sequence');
L = length(x);
h = zeros(1,L);
for i = 1:L
  h(L-i+1) = x(i);
end
N = 2*L-1;
Rxx = zeros(1,N);
for i = L+1:N
   h(i) = 0;
end
for i = L+1:N
    x(i) = 0;
end

for n = 1:N
  for k = 1:N
    if(n >= k)
      Rxx(n) = Rxx(n)+x(n-k+1)*h(k);
    end
  end
end
disp(Rxx,'Auto Correlation Result is')
//Result
//Enter the given discrete time sequence [1 0 0 2]
// Auto Correlation Result is
// Rxx  =    2.    0.    0.    5.    0.    0.    2.
```

Tutorial 4

```
//Caption: Program to find the Cross-correlation of a given Input Sequences
clear all;
clc;
x = [1,2,1,1];
L = length(x);
h1 = [1,1,2,1];
for i = 1:L
  h(L-i+1) = h1(i);
end
N = 2*L-1;
Rxy = zeros(1,N);
for i = L+1:N
   h(i) = 0;
end
for i = L+1:N
    x(i) = 0;
end

for n = 1:N
  for k = 1:N
    if(n >= k)
      Rxy(n) = Rxy(n)+x(n-k+1)*h(k);
    end
  end
end
disp(Rxy,'Cross Correlation Result is')
//Result
//Cross Correlation Result is
//    1.    4.    6.    6.    5.    2.    1
```

Tutorial 5

```
//Caption: Program to find the Circular Convolution of given discrete sequences
clear all;
clc;
x1 = [1,3,5,7];
x2 = [2,4,6,8];
m = length(x1)
n = length(x2)
if (m >n)
   for i = n+1:m
     x2(i) = 0;
   end
elseif (n>m)
  for i = m+1:n
    x1(i) = 0;
  end
end
N = length(x1)
x3 = zeros(1,N);
a(1) = x2(1);
for j = 2:N
  a(j) = x2(N-j+2);
end
for i =1:N
  x3(1) = x3(1)+x1(i)*a(i);
end
for k = 2:N
 for j =2:N
   x2(j) = a(j-1);
 end
   x2(1) = a(N);
   x2
 for i = 1:N
    a(i) = x2(i);
    x3(k) = x3(k)+x1(i)*a(i);
 end
end
disp(x3,'Circular Convolution of Discrete Sequence is x3 = ')
//RESULT
//Circular Convolution of Discrete Sequence is x3 =    84.    92.    84.    60.
```

Tutorial 6

```
//Caption: Program to find the linear convolution using Circular Convolution
clear all;
clc;
x1 = [1,2,3,4];
x2 = [1,1,1,1];
N1 = length(x1)
N2 = length(x2)
N = N1+N2-1;
x1 = [x1,zeros(1,N-N1)];
x2 = [x2,zeros(1,N-N2)];
x3 = zeros(1,N);
a(1) = x2(1);
for j = 2:N
  a(j) = x2(N-j+2);
end
for i =1:N
  x3(1) = x3(1)+x1(i)*a(i);
end
for k = 2:N
 for j =2:N
   x2(j) = a(j-1);
 end
   x2(1) = a(N);
   x2
 for i = 1:N
    a(i) = x2(i);
    x3(k) = x3(k)+x1(i)*a(i);
 end
end
disp(x3,'Linear Convolution using Circular Convolution result is x3 = ')
//RESULT
//Linear Convolution using Circular Convolution result is x3 =
//
//    1.    3.    6.    10.    9.    7.    4.
```

Tutorial 7

```
//Caption: Program to find 8-point DFT using FFT
clear all;
clc;
x = [1,-1,-1,-1,1,1,1,-1];
X = fft(x,-1);
disp(X,'The DFT of x[n] is X(K)=')
//Result
//The DFT of x[n] is X(K)=
//column 1 to 5
//
//    0  - 1.4142136 + 3.4142136i    2. - 2.i    1.4142136 - 0.5857864i    4.
//
//         column 6 to 8
//
//    1.4142136 + 0.5857864i    2. + 2.i  - 1.4142136 - 3.4142136i
```

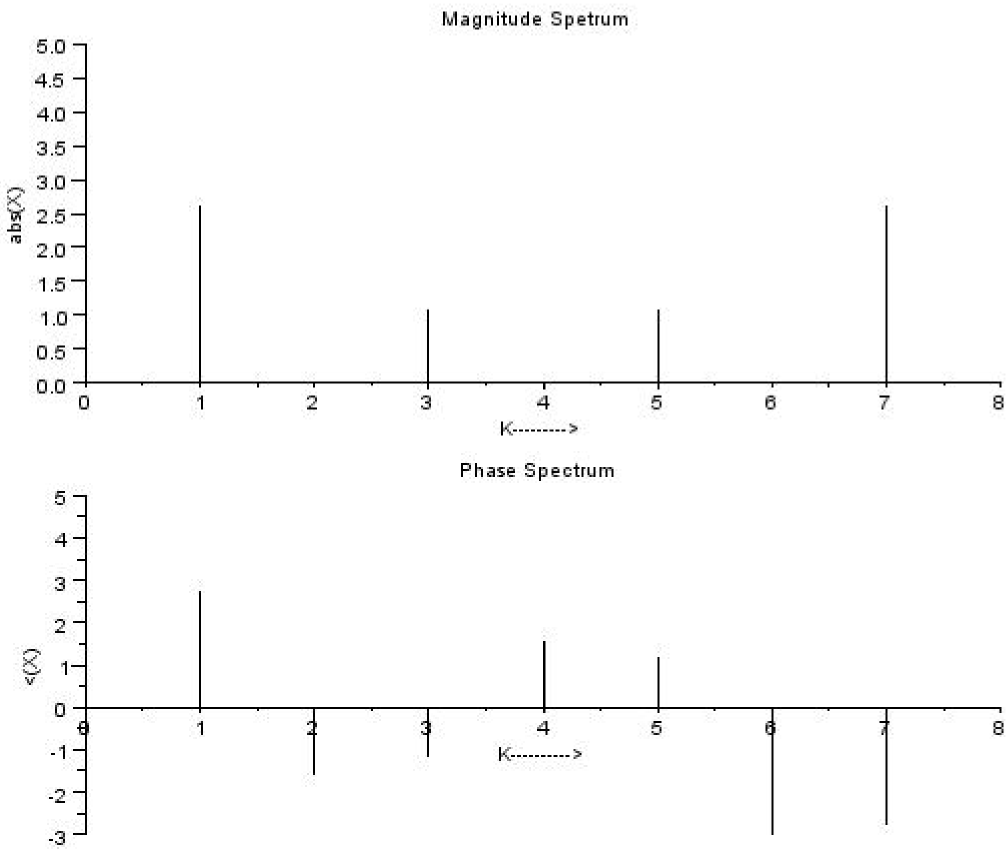
Tutorial 8

```
//Caption: Program to find 8-point IDFT using FFT
clear all;
clc;
X=[0,-1.4142136+3.4142136*%i,2-2*%i,1.4142136-0.5857864*%i,4,
1.4142136+0.5857864*%i, 2+2*%i,-1.4142136-3.4142136*%i];
x = fft(X,1);
disp(x,'The IDFT of X[K] is x[n]=')
//Result
//The IDFT of X[K] is x[n]=
//1.  - 1.  - 1.  - 1.    1.    1.    1.  - 1.
```

Tutorial 9

```
//Caption:Determination of N-point DFT
//Plot its magnitude and phase spectrum
clear all;
clc;
L = 8;  // Length of the sequence
N = 8;  // N -point DFT
x = [0,0,0,1,1,1,1,0] //input discrete sequence
n = 0:N-1;
//Computing DFT
X = fft(x,-1)
K = 0:N-1;
Phase_X = atan(imag(X),real(X));
subplot(2,1,1)
a =gca();
a.x_location ='origin';
a.y_location ='origin';
a.data_bounds = [0,0;8,5]
plot2d3('gnn',K,abs(X))
xlabel('K--------->')
ylabel('abs(X)')
title('Magnitude Spetrum')
subplot(2,1,2)
a =gca();
a.x_location ='origin';
a.y_location ='origin';
```

```
a.data_bounds = [0,0;8,5]
plot2d3('gnn',K,Phase_X)
xlabel('K---------->')
ylabel('<(X)')
title('Phase Spectrum')
```



Tutorial 10

```
//Caption: Program to find the transfer function using the poles and zeros given
//Transfer function
clear all;
clc;
S = poly(0,'S');
pol = [-0.3+%i*0.4,-0.3-%i*0.4];
zero = -0.2;
H = (S-zero)/((S-pol(1))*(S-pol(2)));
disp(H,'H(S)=')
//Result
//H(S)=
//
//
//        0.2 + S
//   --------------------
//                    2
//   0.25 +  0.6S +  1S
```